



Performance Tuning for Intel® Xeon Phi™ Coprocessors

Robert Reed
Intel Technical Consulting Engineer

Agenda

Start tuning on host

Overview of Intel® VTune™ Amplifier XE

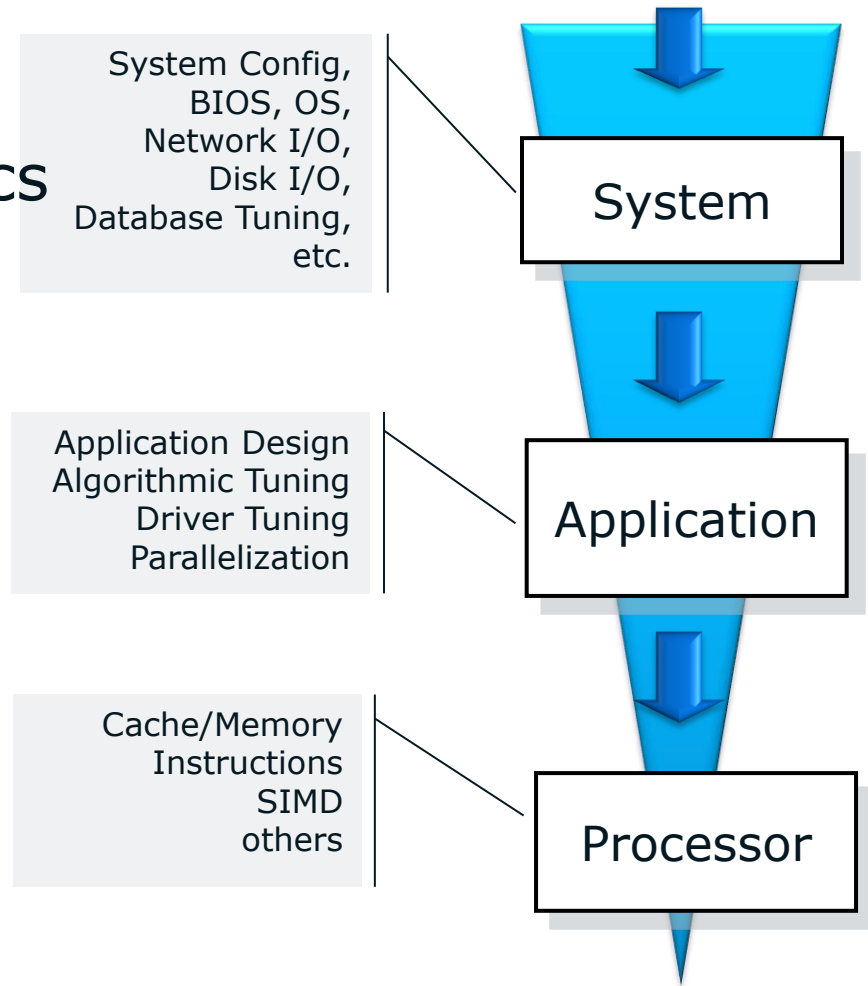
Efficiency metrics

Problem areas

Performance Analysis Methodology

Optimization: A Top-down Approach

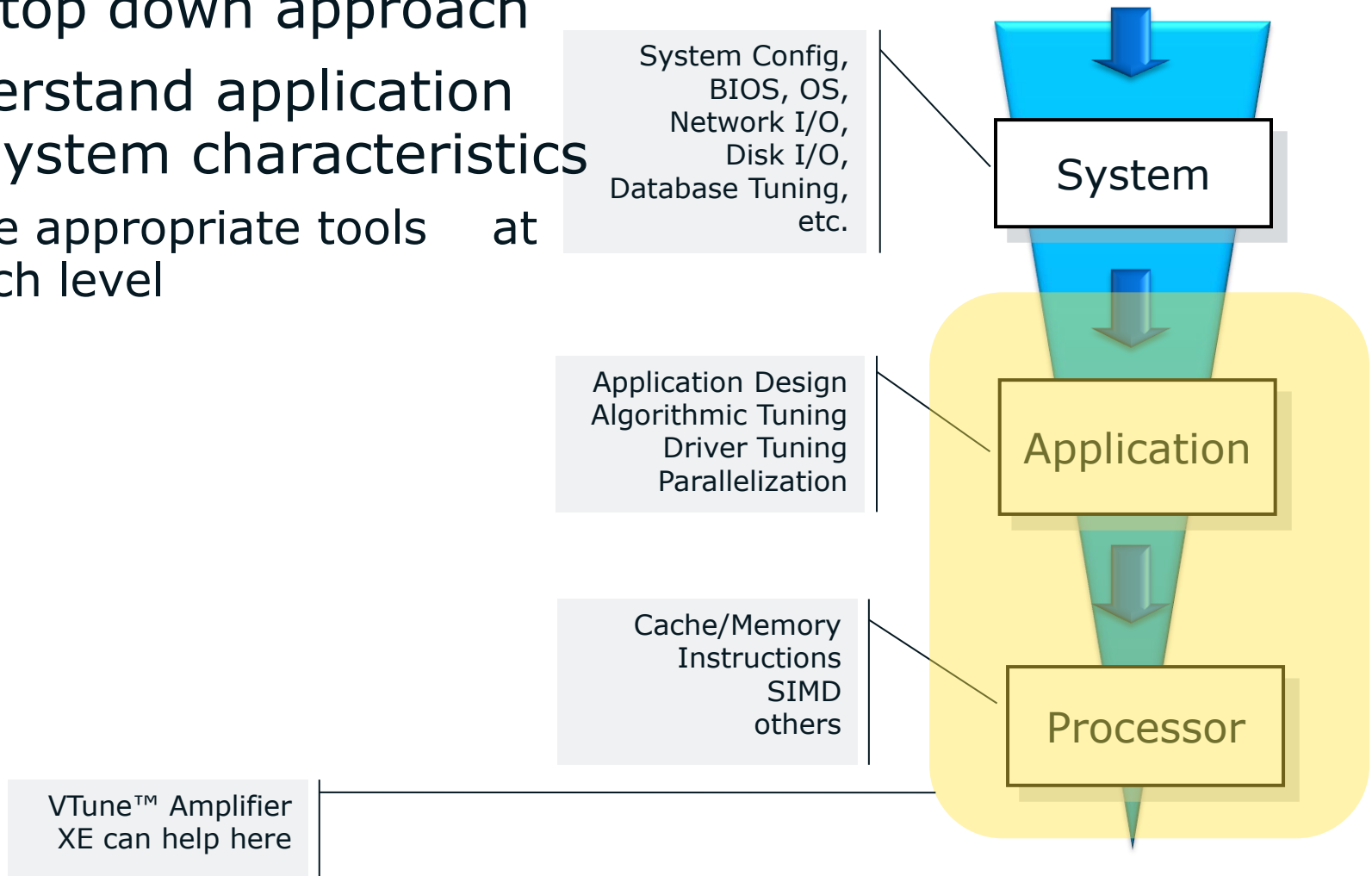
- Use top down approach
- Understand application and system characteristics
 - Use appropriate tools at each level



Performance Analysis Methodology

Optimization: A Top-down Approach

- Use top down approach
- Understand application and system characteristics
 - Use appropriate tools at each level



Start with **host-based** profiling to identify vectorization/ parallelism/ offload candidates

Start with representative/reasonable workloads!

Use Intel® VTune™ Amplifier XE to gather hot spot data

- Tells what functions account for most of the run time
- Often, this is enough
 - But it does not tell you much about program structure

Start with **host-based** profiling to identify vectorization/ parallelism/ offload candidates

Start with representative/reasonable workloads!

Use Intel® VTune™ Amplifier XE to gather hot spot data

- Tells what functions account for most of the run time
- Often, this is enough
 - But it does not tell you much about program structure

Alternately, profile functions & loops using Intel® Composer XE

- Build with options
`-profile-functions -profile-loops=all -profile-loops-report=2`
- Run the code (which may run slower) to collect profile data
- Look at the resulting `dump` files, or open the `xml` file with the data viewer `loopprofileviewer.sh` located in the compiler `./bin` directory
- Tells you
 - which loops and functions account for the most run time
 - how many times each loop executes (min, max and average)

Correctness/Performance Analysis of Parallel code

Intel® Inspector XE and thread-reports in VTune™ Amplifier XE are not available on the Intel® Xeon Phi™ coprocessor

So...

Correctness/Performance Analysis of Parallel code

Intel® Inspector XE and thread-reports in VTune™ Amplifier XE are not available on the Intel® Xeon Phi™ coprocessor

So...

- Use Intel Inspector XE on your code with **offload disabled** (on host) to identify correctness errors (e.g., deadlocks, races)
 - Once fixed, then enable offload and continue debugging on the coprocessor

Correctness/Performance Analysis of Parallel code

Intel® Inspector XE and thread-reports in VTune™ Amplifier XE are not available on the Intel® Xeon Phi™ coprocessor

So...

- Use Intel Inspector XE on your code with **offload disabled** (on host) to identify correctness errors (e.g., deadlocks, races)
 - Once fixed, then enable offload and continue debugging on the coprocessor
- Use VTune Amplifier XE's parallel performance analysis tools to find issues on the host by running your program with **offload disabled**
 - Fix everything you can
 - Then study scaling on the coprocessor using lessons from host tuning to further optimize parallel performance
 - Be wary of synchronization across more than a handful of threads
 - Pay attention to load balance.

Agenda

Start tuning on host

Overview of Intel® VTune™ Amplifier XE

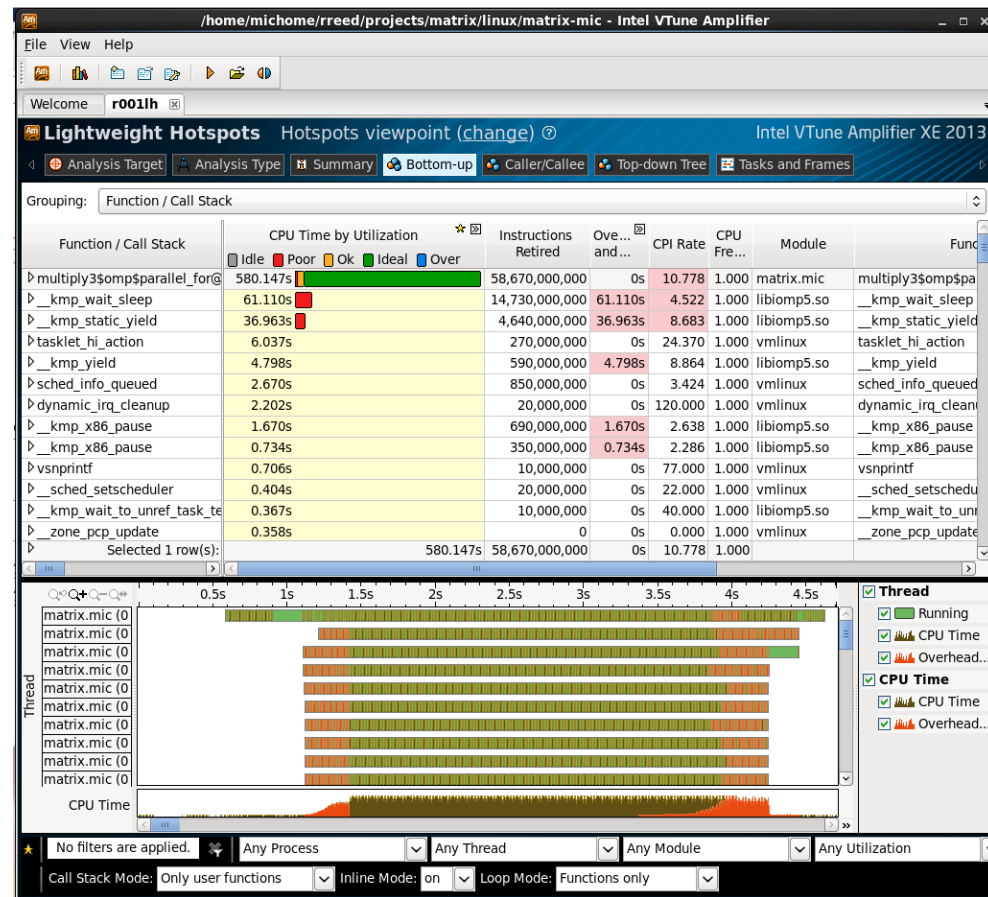
Efficiency metrics

Problem areas

Intel® VTune™ Amplifier XE

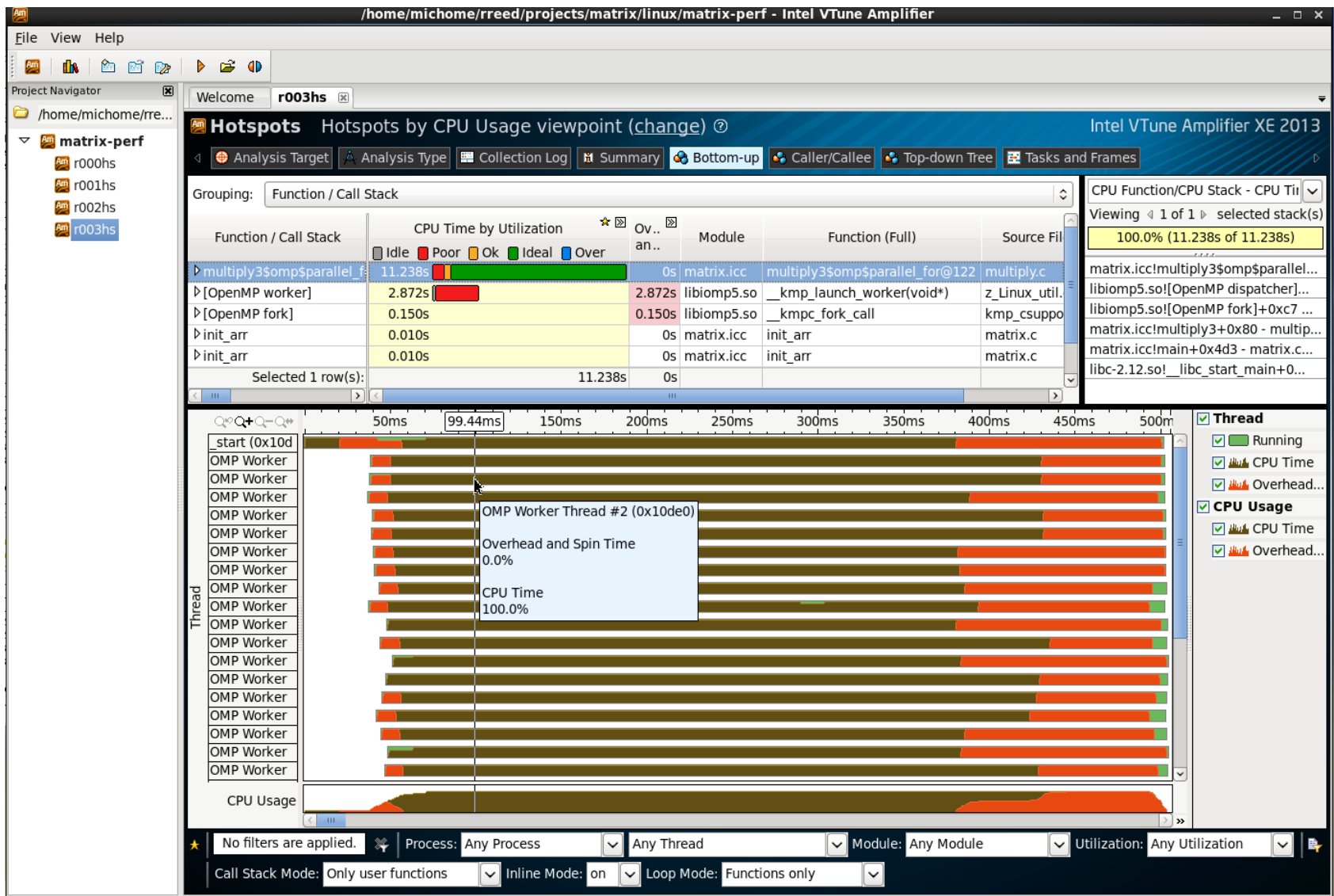
Tune Applications for Scalable Multicore Performance

- **Fast, Accurate Performance Profiles**
 - Hotspot (Statistical call tree)
 - Hardware-Event Based Sampling
- **Thread Profiling**
 - Visualize thread interactions on timeline
 - Balance workloads
- **Easy set-up**
 - Pre-defined performance profiles
 - Use a normal production build
- **Compatible**
 - Microsoft*, GCC*, Intel compilers
 - C/C++, Fortran, Assembly, .NET*
 - Latest Intel processors and compatible processors¹
- **Find Answers Fast**
 - Filter out extraneous data
 - View results tied to source/assembly lines
 - Event multiplexing
- **Windows* or Linux***
 - Visual Studio* Integration (Windows)
 - Standalone user interface and command line
 - 32 and 64-bit

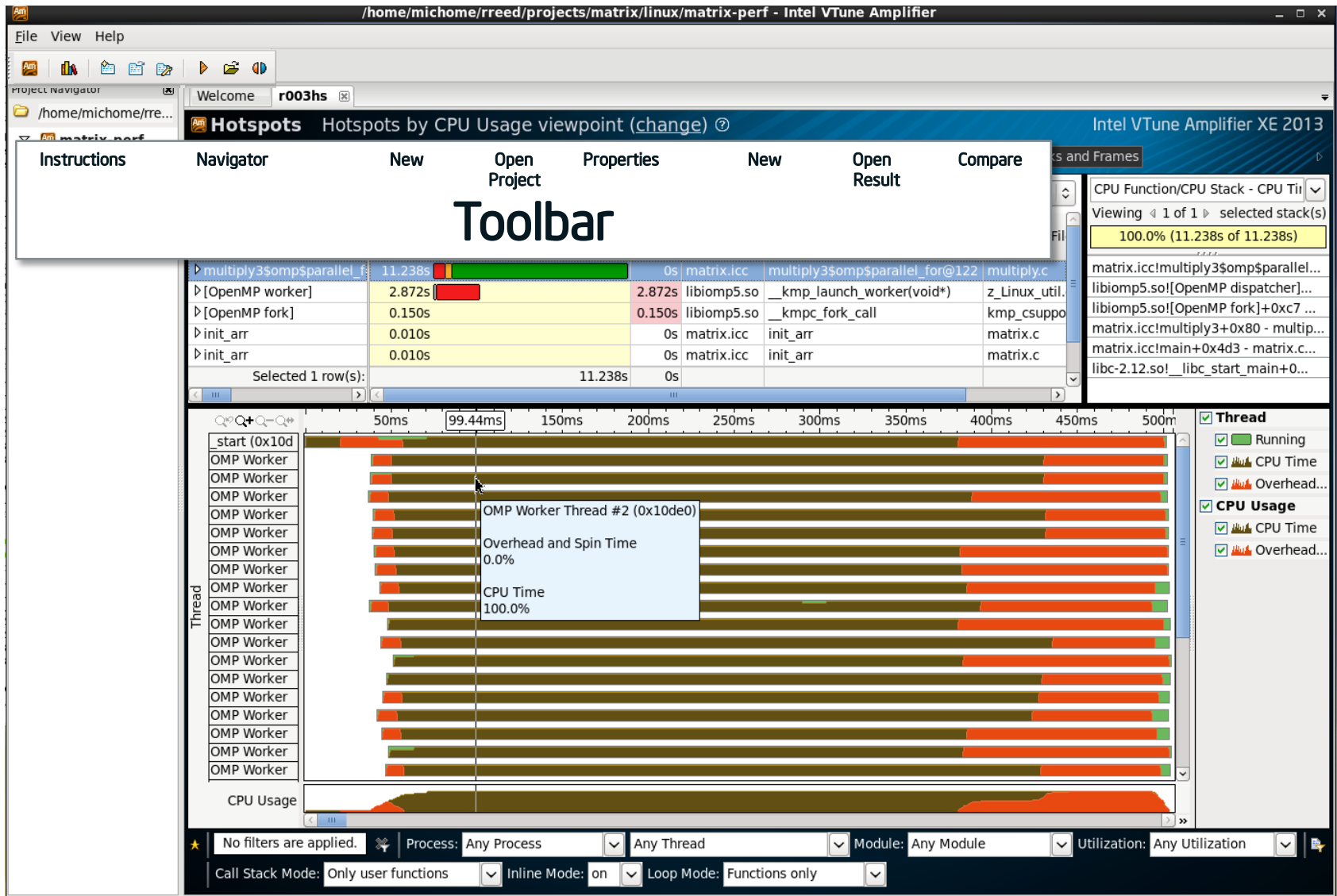


¹ IA-32 and Intel® 64 architectures.
Many features work with compatible processors.
Event based sampling requires a genuine Intel Processor.

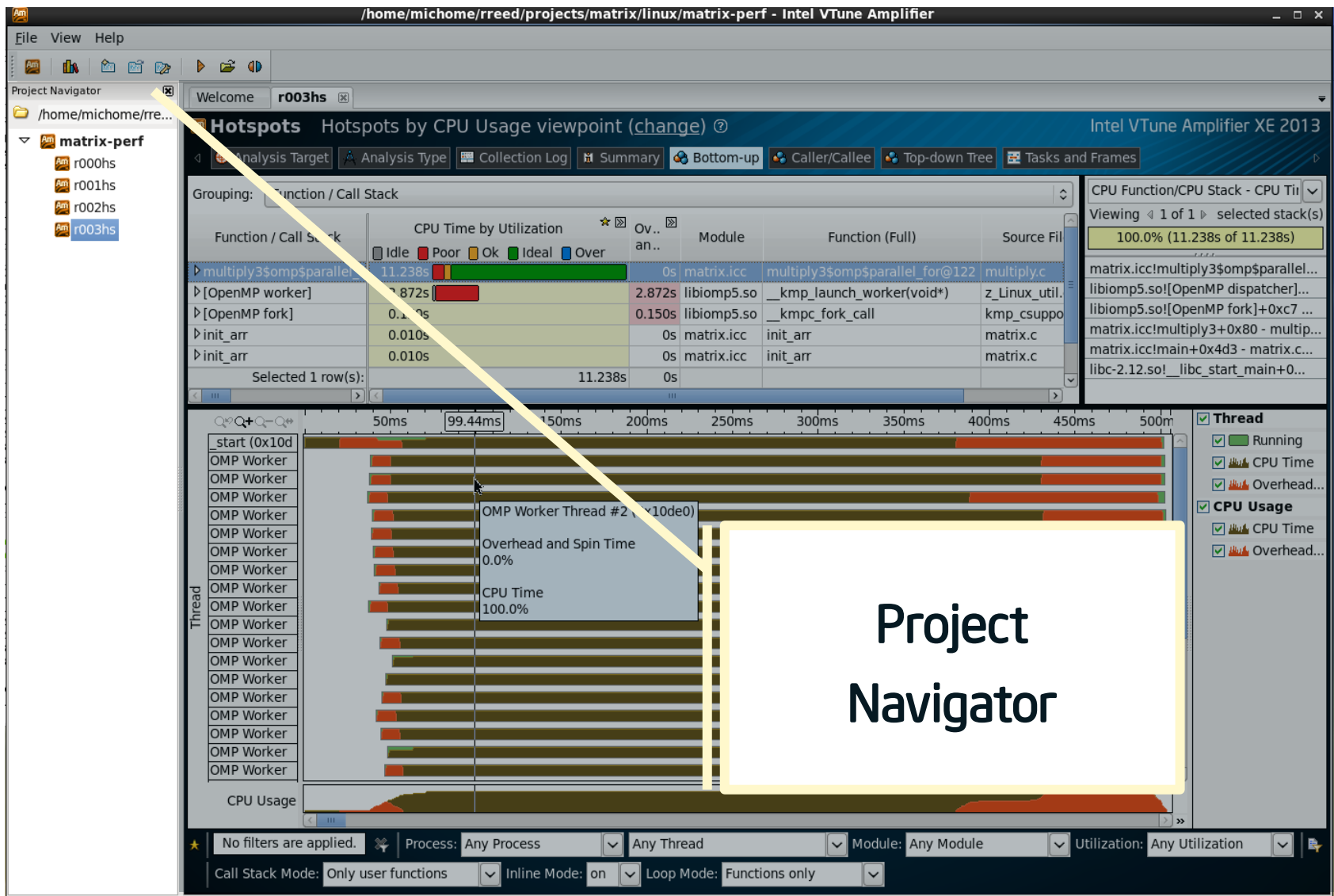
VTune™ Amplifier XE visualizes performance



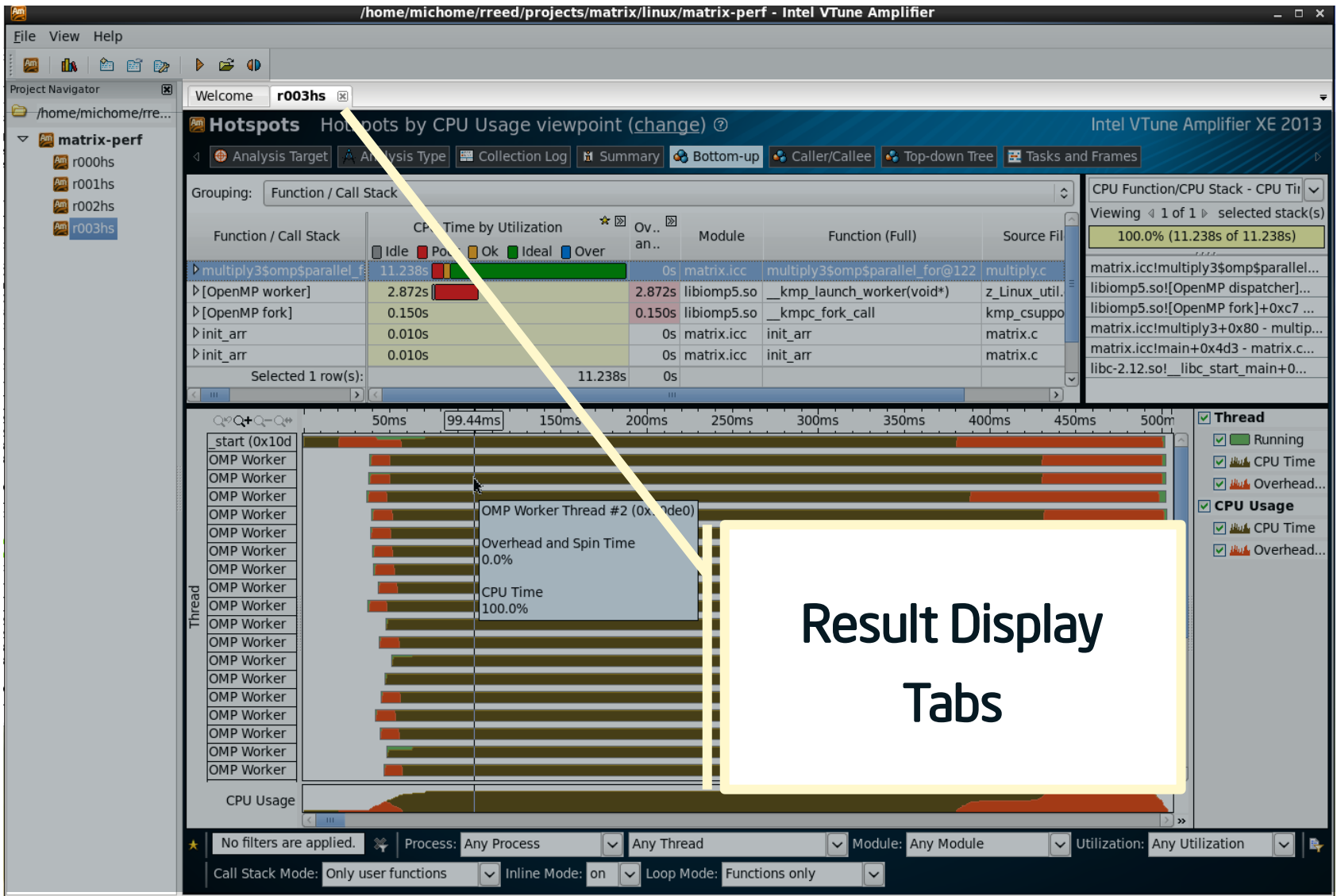
VTune™ Amplifier XE visualizes performance



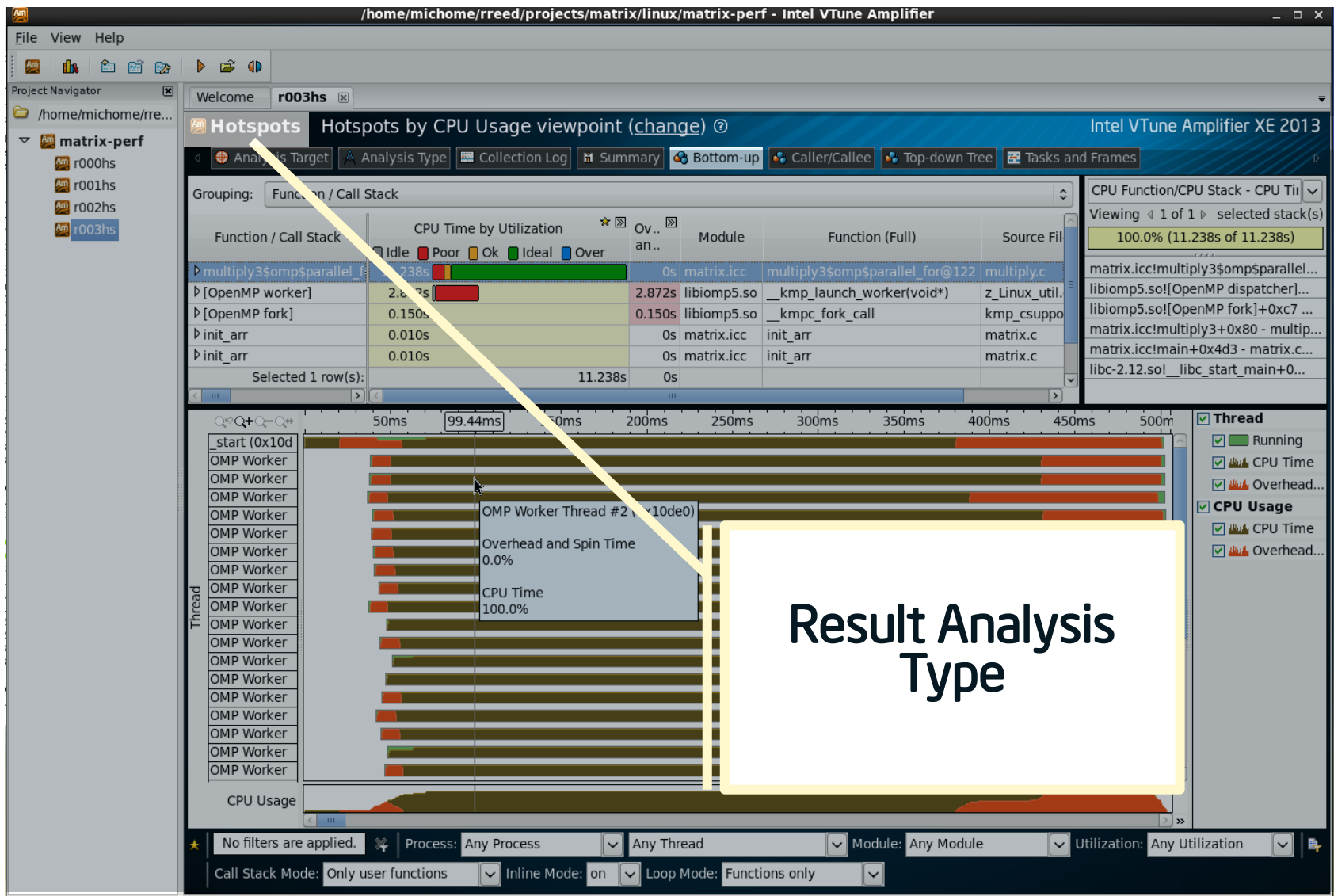
VTune™ Amplifier XE visualizes performance



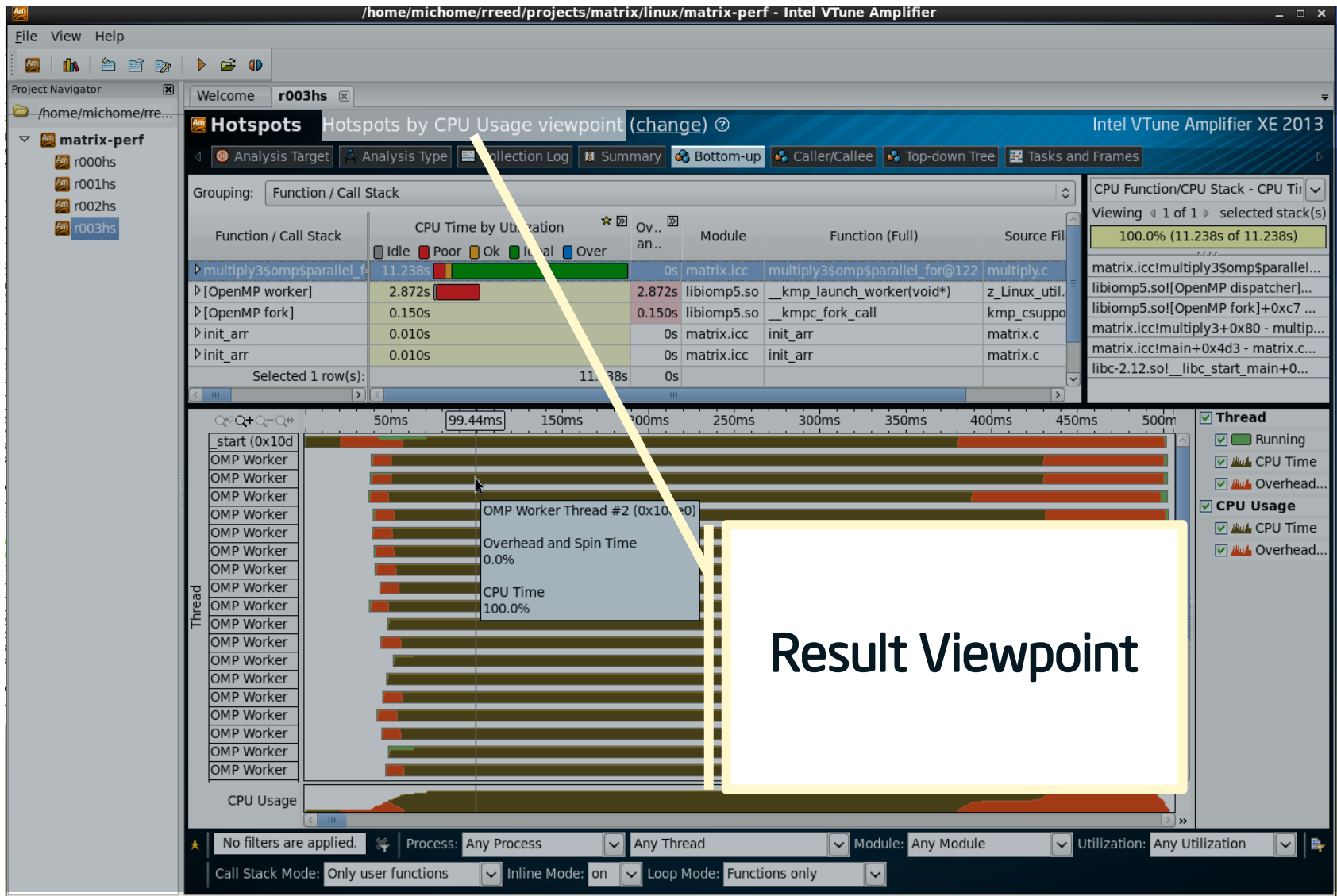
VTune™ Amplifier XE visualizes performance



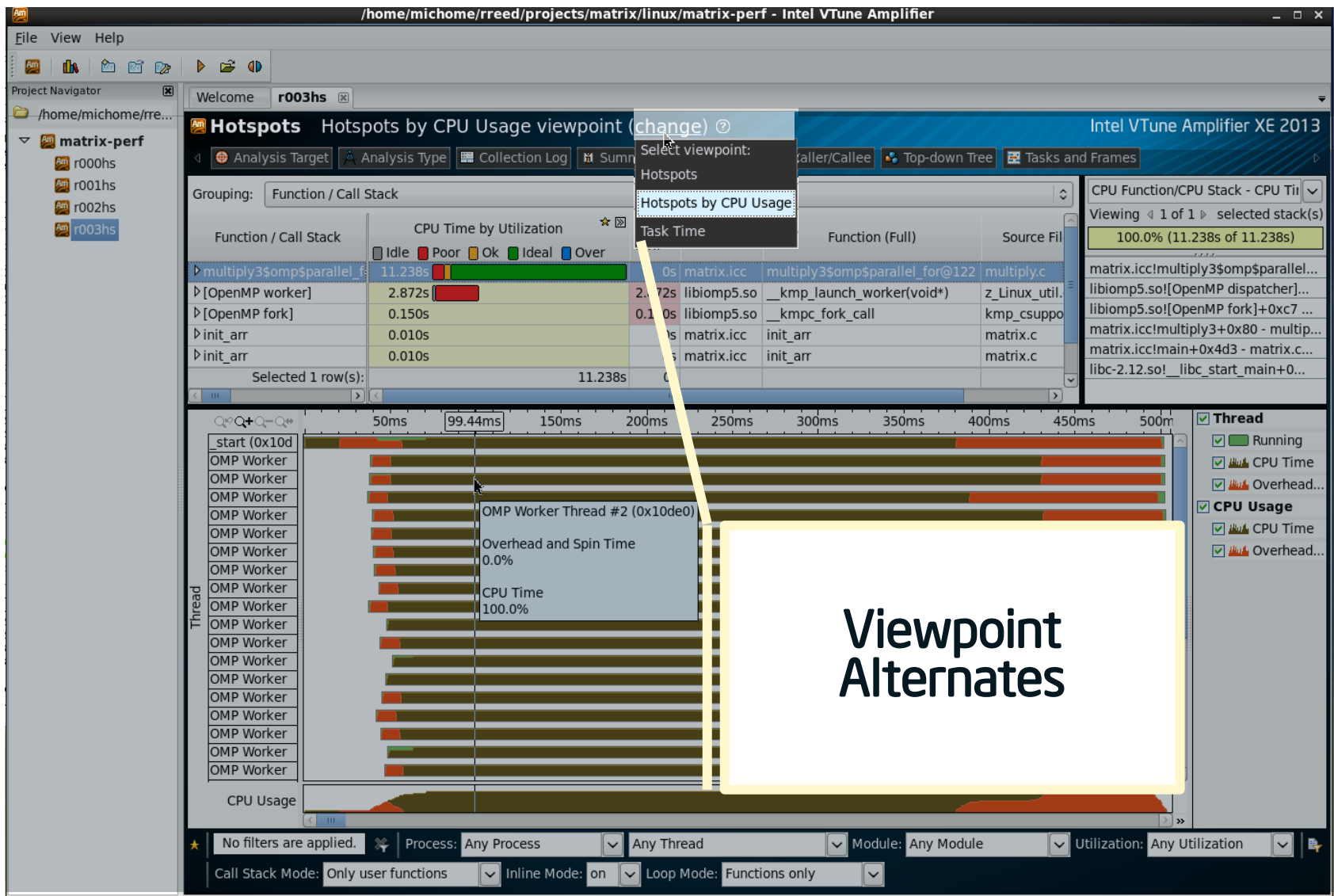
VTune™ Amplifier XE visualizes performance



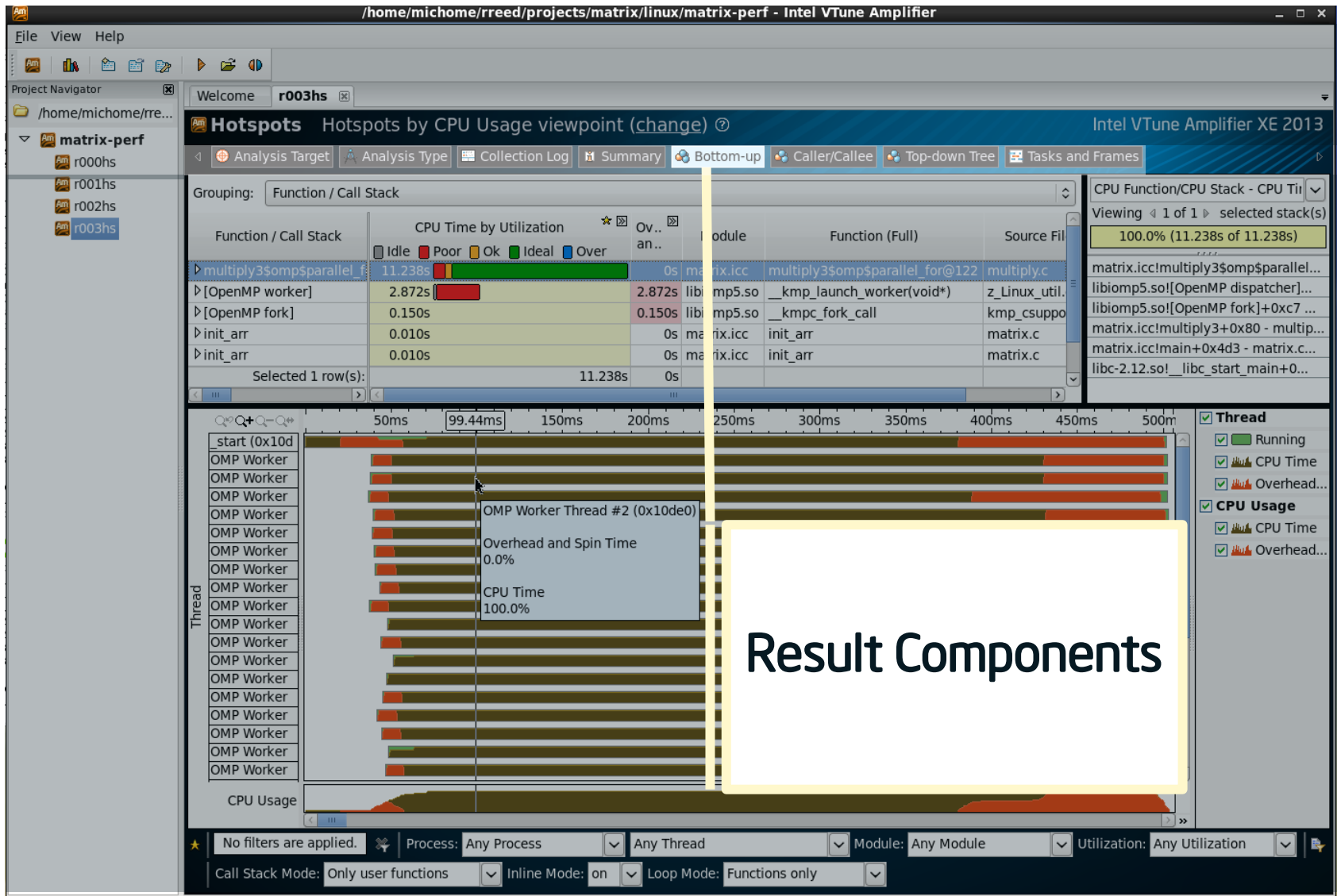
VTune™ Amplifier XE visualizes performance



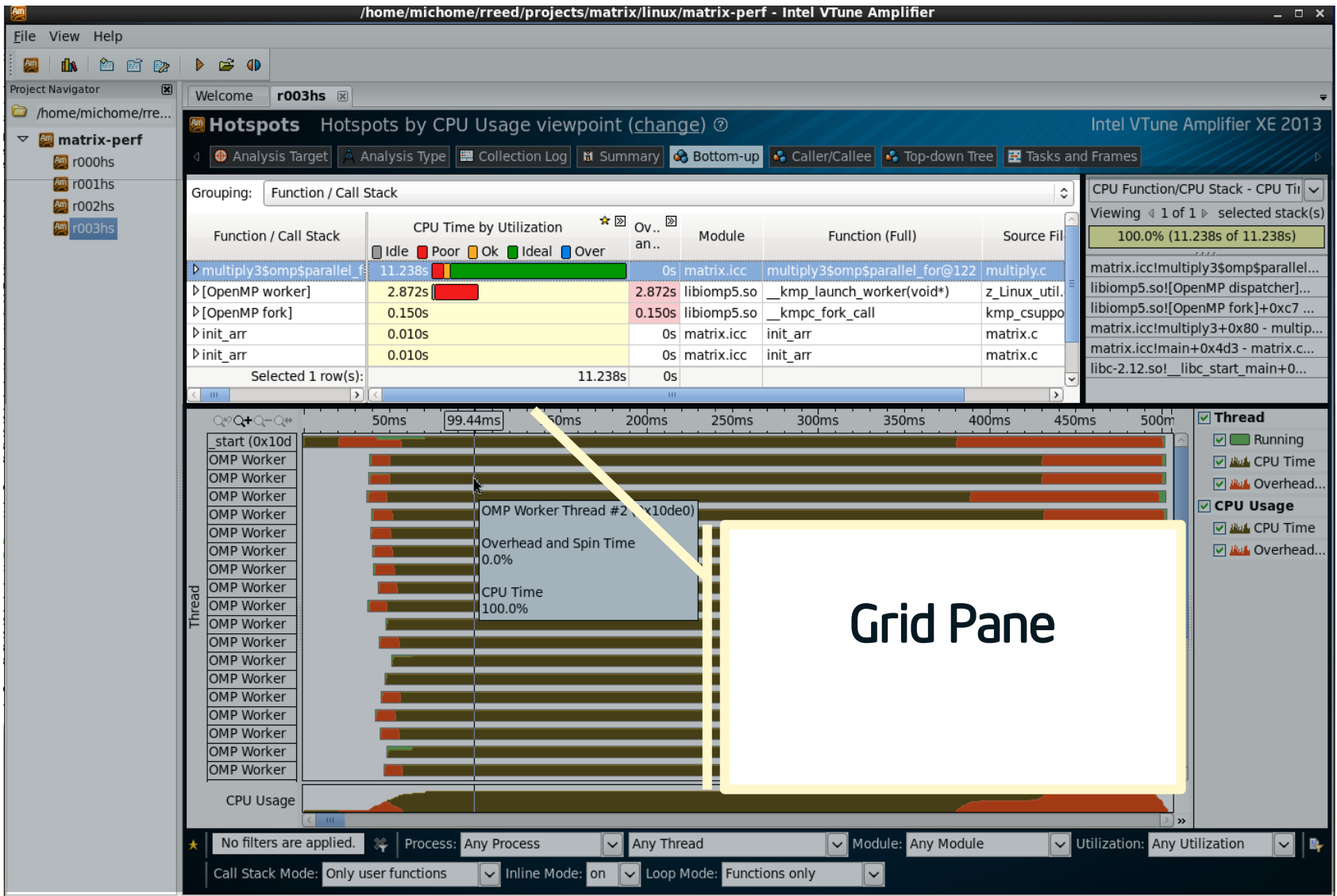
VTune™ Amplifier XE visualizes performance



VTune™ Amplifier XE visualizes performance



VTune™ Amplifier XE visualizes performance

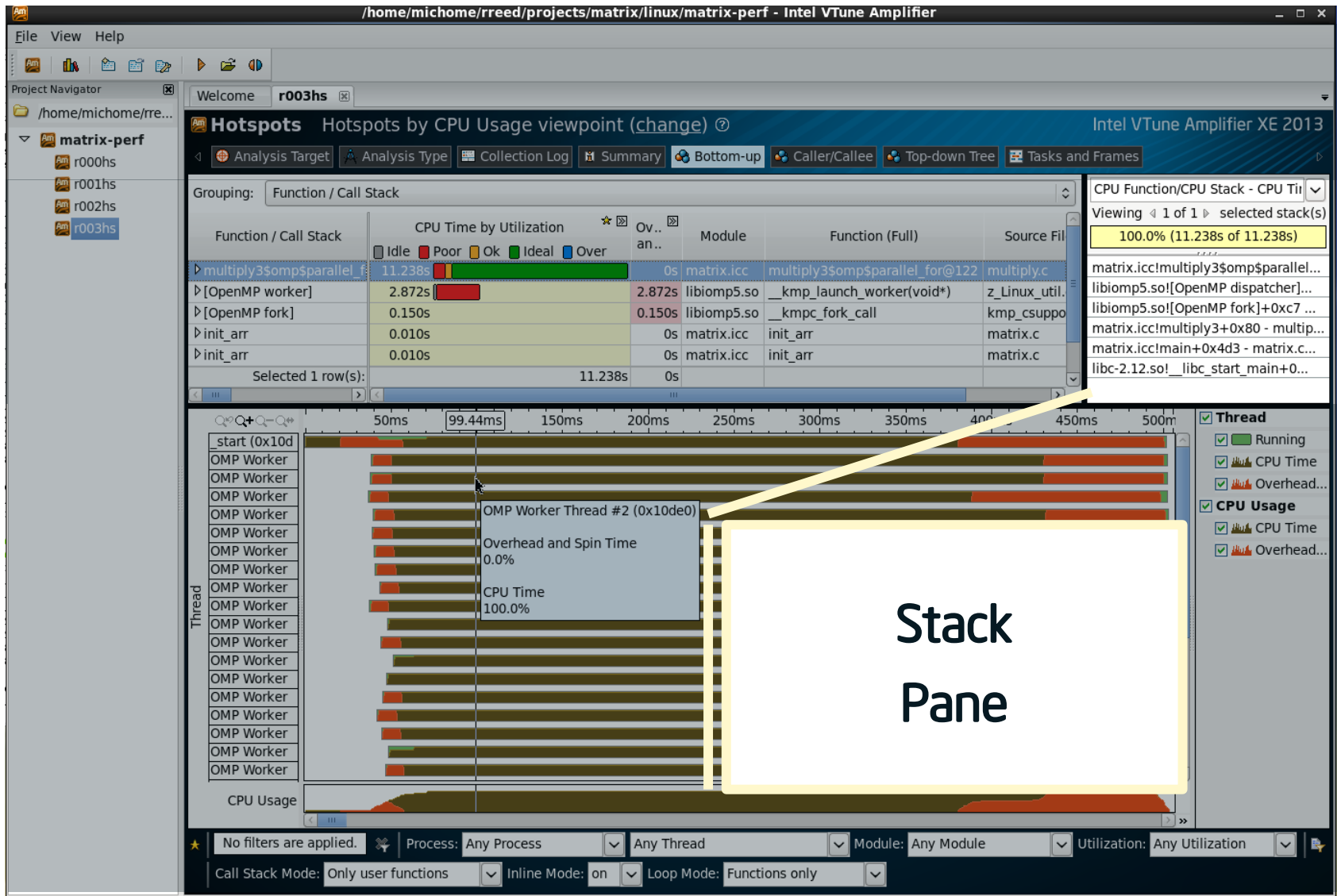


VTune™ Amplifier XE visualizes performance

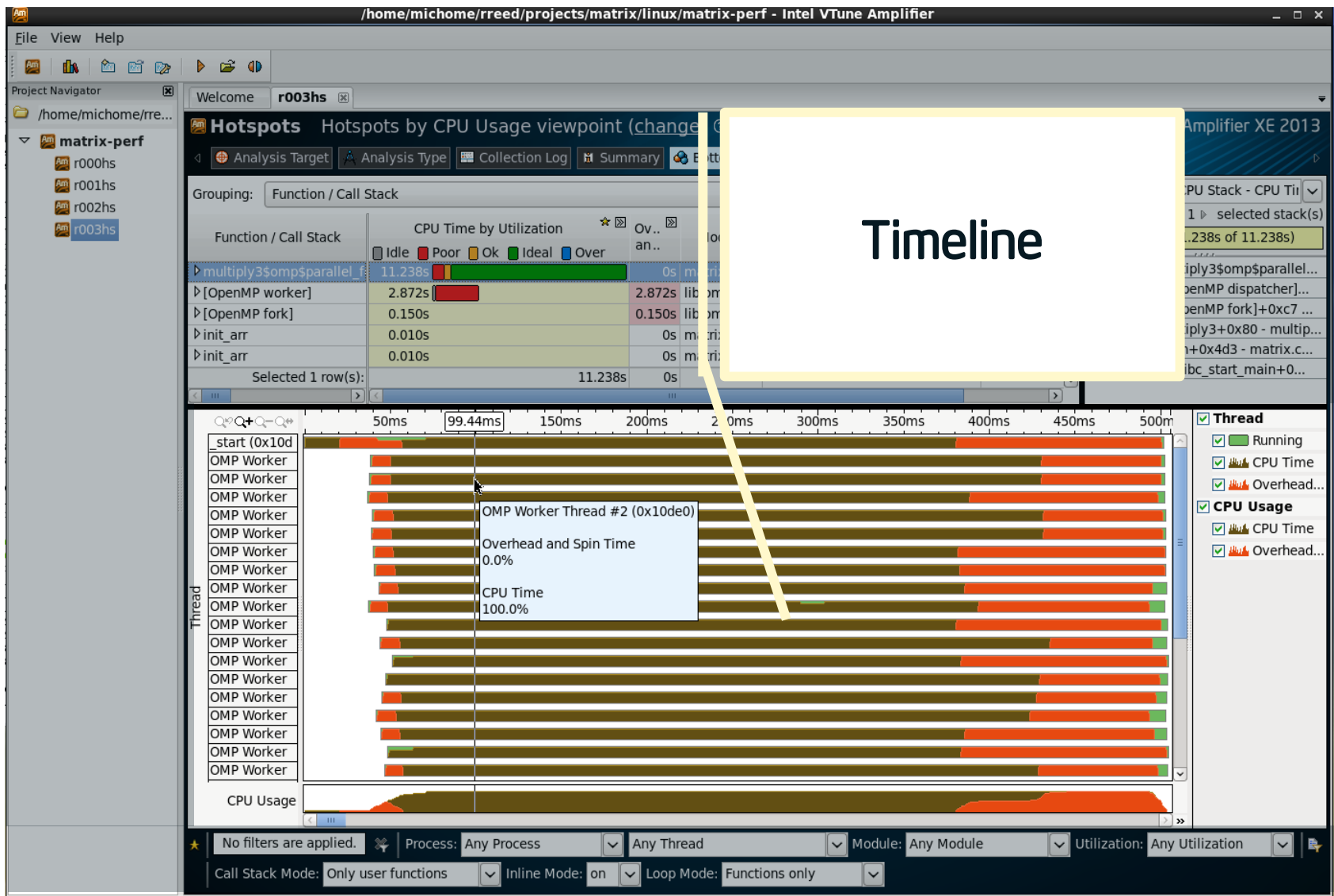
The screenshot displays the Intel VTune Amplifier XE 2013 interface. The main window is titled "Hotspots by CPU Usage viewpoint (change)". The left sidebar shows a project named "matrix-perf" with several sub-items, including "r003hs" which is selected. The main pane shows a list of hotspots, with "multiply3s" selected. A yellow box highlights the "Grouping:" pull-down menu, which is open, showing a list of grouping options such as "Function / Call Stack", "Source Function / Function / Call Stack", "Module / Function / Call Stack", etc. A yellow arrow points from the text "Grid Pane" and "Grouping pull-down" to the pull-down menu. The right pane shows the "CPU Function/CPU Stack - CPU Ti" view, displaying a stack of functions and their CPU time. The bottom status bar shows various filters and modes, including "Call Stack Mode: Only user functions", "Inline Mode: on", and "Loop Mode: Functions only".

Grid Pane
Grouping pull-down

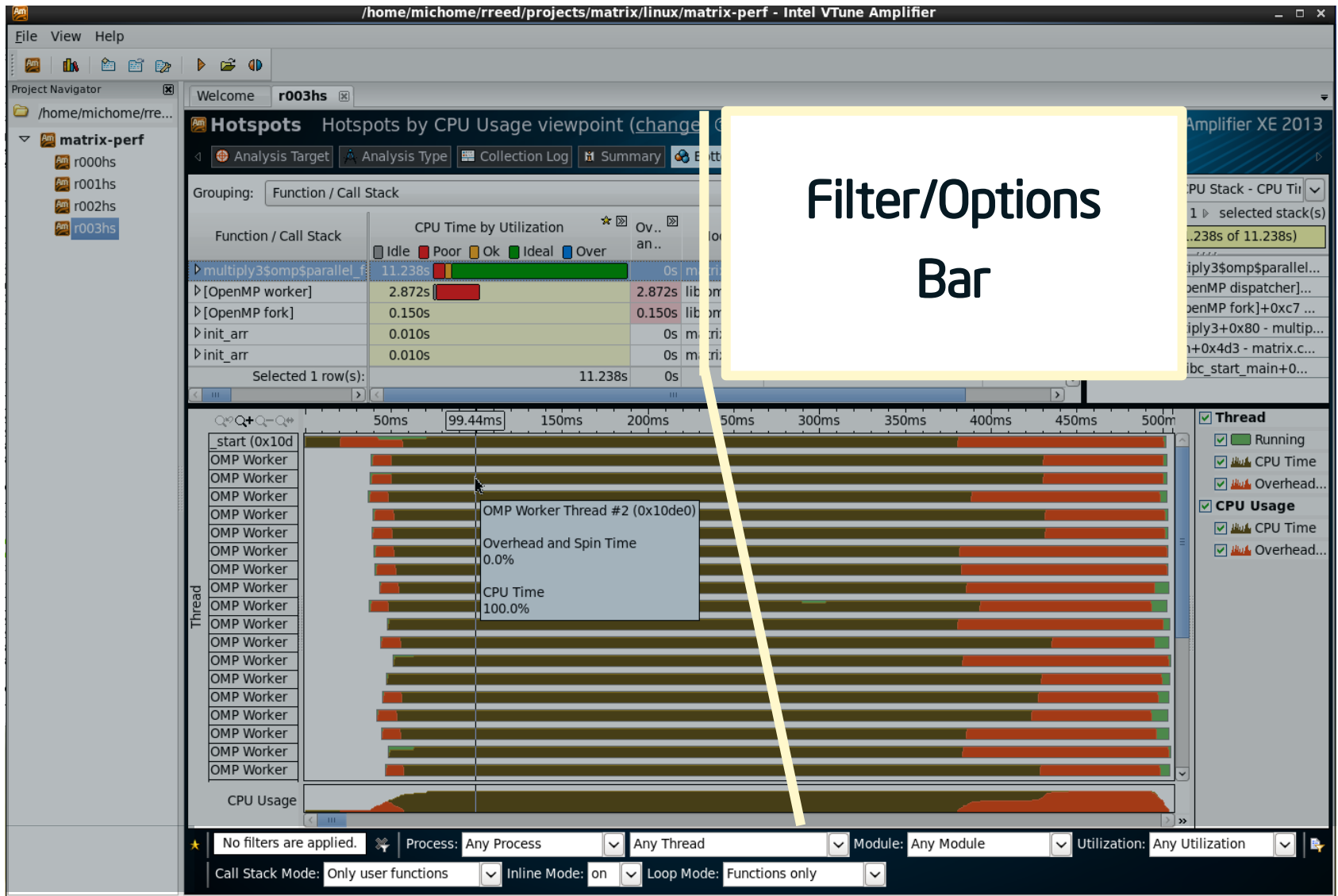
VTune™ Amplifier XE visualizes performance



VTune™ Amplifier XE visualizes performance



VTune™ Amplifier XE visualizes performance



VTune™ Amplifier XE visualizes performance

Source View / Per line localization

The screenshot displays the Intel VTune Amplifier XE 2013 interface. The main window is titled "/home/michome/rreed/projects/matrix/linux/matrix-mic - Intel VTune Amplifier". The "Lightweight Hotspots" tab is active, showing a "Hotspots viewpoint (change)". The "Analysis Target" is "r001lh". The "Analysis Type" is "Summary". The "Bottom-up" view is selected. The "Source" tab is active, showing C code for a matrix multiplication. The code is as follows:

```
113 for (k = k0; k < k0 + mblock; k++) {
114     #pragma unroll(8)
115     #pragma ivdep
116     for (j = j0; j < j0 + mblock; j++) {
117         c[i][j] = c[i][j] + a[i][k] * b[k][j];
118     }
119 }
120 }
121 }
122 }
123 }
124 #elif defined (USE_OMP) // {}
125 #pragma omp parallel for collapse (2)
126 for(i=0; i<msize; i++) {
127     for(k=0; k<msize; k++) {
128         #pragma unroll(8)
129         #pragma ivdep
130         for(j=0; j<msize; j++) {
131             c[i][j] = c[i][j] + a[i][k] * b[k][j];
132         }
133     }
134 }
135 #endif // }
136 }
137 }
```

The assembly view on the right shows the corresponding assembly instructions and their performance metrics. The assembly is as follows:

Address	Basic Block / Address	Function Range / Basic Block	Rate	Time by Utilization
0x402a94	131	vprefetch0 (%rcx)	0.183s	
0x402a98	131	vbroadcastsq (%r9,%rbx,8), %k0, %zmm0	2.954s	
0x402a9f	131	vprefetch0 0x200(%rcx)	0.174s	
0x402aa6	131	vprefetch0 (%r8)	0.294s	
0x402aab	130	movsxd %edx, %r9	0.018s	
0x402aae	131	vprefetch0 0x200(%r8)	0.073s	
0x402ab5	131	mov %al, %al	0.009s	
0x402ab7	131	vprefetch0 0x40(%rcx)	0.248s	
0x402abc	131	mov %al, %al	0.009s	
0x402abe	131	vprefetch0 0x240(%rcx)	0.257s	
0x402ac5	Block 24:			
0x402ac5	131	vmovapd (%r8,%rax,8), %k0, %zmm1	6.147s	
0x402acc	131	vprefetch1 0x1000(%rcx,%rax,8)	11.817s	
0x402ad4	131	vmovapd 0x40(%r8,%rax,8), %k0, %zmm2	6.982s	
0x402adc	131	vprefetch0 0x400(%rcx,%rax,8)	9.000s	
0x402ae4	131	vmovapd 0x80(%r8,%rax,8), %k0, %zmm3	6.596s	
0x402aec	131	vprefetch1 0x1000(%r8,%rax,8)	13.835s	
0x402af4	131	vmovapd 0xc0(%r8,%rax,8), %k0, %zmm4	6.917s	
0x402afc	131	vprefetch0 0x400(%r8,%rax,8)	8.486s	

The bottom of the window shows the "Selected 1 row(s): 528.3" and "Selected 0 row(s):". The bottom status bar shows "No filters are applied." and "Process: Any Process", "Thread: Any Thread", "Module: Any Module", "Utilization: Any Utilization". The "Call Stack Mode" is "Only user functions", "Inline Mode" is "on", and "Loop Mode" is "Functions only".

VTune™ Amplifier XE visualizes performance

The screenshot displays the Intel VTune Amplifier XE 2013 interface. The main window is titled "/home/michome/rreed/projects/matrix/linux/matrix-mic - Intel VTune Amplifier". The "Lightweight Hotspots" tab is active, showing a "Hotspots viewpoint (change)". The "Analysis Target" is "r001lh". The "Analysis Type" is "Summary". The "Bottom-up" view is selected. The "Source" tab is active, showing C code for a matrix multiplication function. The code includes pragmas for unroll and ivdep, and a parallel region for collapse (2). The assembly view on the right shows the corresponding instructions, including prefetch and broadcast instructions. A callout box with a yellow border highlights the "Source View / View / Hot spot Navigation controls" area, which includes a dropdown menu for "Basic Block / Address" and a "Function Range / Basic Block" field. The bottom status bar shows "No filters are applied" and various filter settings for Process, Thread, Module, and Utilization.

Source View /
View / Hot spot
Navigation controls

Source

```
113     for (k = k0; k < k0 + mblock; k++) {  
114     #pragma unroll(8)  
115     #pragma ivdep  
116         for (j = j0; j < j0 + mblock; j++) {  
117             c[i][j] = c[i][j] + a[i][k] * b[k][j];  
118         }  
119     }  
120 }  
121 }  
122 }  
123 }  
124 #elif defined (USE_OMP) // {}  
125     #pragma omp parallel for collapse (2)  
126     for(i=0; i<msize; i++) {  
127         for(k=0; k<msize; k++) {  
128             #pragma unroll(8)  
129             #pragma ivdep  
130             for(j=0; j<msize; j++) {  
131                 c[i][j] = c[i][j] + a[i][k] * b[k][j];  
132             }  
133         }  
134     }  
135 #endif // }  
136 }  
137 }
```

Selected 1 row(s): 528.3

Selected 0 row(s):

No filters are applied. Process: Any Process Thread: Any Thread Module: Any Module Utilization: Any Utilization

Call Stack Mode: Only user functions Inline Mode: on Loop Mode: Functions only

VTune™ Amplifier XE visualizes performance

Intel VTune Amplifier XE 2013

Lightweight Hotspots Hotspots viewpoint (change) ?

Analysis Target: r001lh Analysis Type: Summary Bottom-up Caller/Callee Top-down Tree Tasks and Frames multiply.c

Source Assembly

Assembly grouping:

Source

Address

Basic Block / Address

Function Range / Basic Block / Address

Assembly

CPU Time by Utilization

Idle Poor Ok Ideal

0.073s

0.046s

0.009s

0x402a7f 130 add %r9, %r8

0x402a82 130 lea (%r8,%rax,8), %r10

0x402a86 130 and \$0x3f, %r10

0x402a8a 130 jnz 0x402ca2 <Block 26>

0x402a90 Block 23:

0x402a90 131 lea (%rsi,%r13,1), %r9

0x402a94 131 vprefetch0 (%rcx)

0x402a98 131 vbroadcastsq (%r9,%rbx,8), %k0, %zmm0

0x402a9f 131 vprefetch0 0x200(%rcx)

0x402aa6 131 vprefetch0 (%r8)

0x402aab 130 movsxd %edx, %r9

0x402aae 131 vprefetch0 0x200(%r8)

0x402ab5 131 mov %al, %al

0x402ab7 131 vprefetch0 0x40(%rcx)

0x402abc 131 mov %al, %al

0x402abe 131 vprefetch0 0x240(%rcx)

0x402ac5 Block 24:

0x402ac5 131 vmovapd (%r8,%rax,8), %k0, %zmm1

0x402acc 131 vprefetch1 0x1000(%rcx,%rax,8)

0x402ad4 131 vmovapd 0x40(%r8,%rax,8), %k0, %zmm2

0x402adc 131 vprefetch0 0x400(%rcx,%rax,8)

0x402ae4 131 vmovapd 0x80(%r8,%rax,8), %k0, %zmm3

0x402aec 131 vprefetch1 0x1000(%r8,%rax,8)

0x402af4 131 vmovapd 0xc0(%r8,%rax,8), %k0, %zmm4

0x402afc 131 vprefetch0 0x400(%r8,%rax,8)

5.780s

0.606s

45.404s

528.349s

Selected 1 row(s): 528.3

Selected 0 row(s):

No filters are applied. Process: Any Process Thread: Any Thread Module: Any Module Utilization: Any Utilization

Call Stack Mode: Only user functions Inline Mode: on Loop Mode: Functions only

Assembly View / View / Hot spot Navigation controls

VTune™ Amplifier XE visualizes performance

Assembly View / Assembly groupings

Analysis Target: multiply.c
Analysis Type: Bottom-up
Summary: Bottom-up
Caller/Callee: Top-down Tree
Tasks and Frames: multiply.c

Source Line	Source	Address	Disassembly	CPU Time by Utilization
113		0x402a7f	add %r9, %r8	0.073s
114		0x402a82	lea (%r8,%rax,8), %r10	0.046s
115		0x402a86	and \$0x3f, %r10	0.009s
116		0x402a8a	jnz 0x402ca2 <Block 26>	
117		0x402a90	Block 23:	
118		0x402a90	lea (%rsi,%r13,1), %r9	0.028s
119		0x402a94	vprefetch0 (%rcx)	0.183s
120		0x402a98	vbroadcastsq (%r9,%rbx,8), %k0, %zmm0	2.954s
121		0x402a9f	vprefetch0 0x200(%rcx)	0.174s
122		0x402aa6	vprefetch0 (%r8)	0.294s
123		0x402aab	movsxd %edx, %r9	0.018s
124		0x402aae	vprefetch0 0x200(%r8)	0.073s
125	#pragma omp parallel for collapse (2)	0x402ab5	mov %al, %al	0.009s
126	for(i=0; i<msize; i++) {	0x402ab7	vprefetch0 0x40(%rcx)	0.248s
127	for(k=0; k<msize; k++) {	0x402abc	mov %al, %al	0.009s
128	#pragma unroll(8)	0x402abe	vprefetch0 0x240(%rcx)	0.257s
129	#pragma ivdep	0x402ac5	Block 24:	
130	for(j=0; j<msize; j++) {	0x402ac5	vmovapd (%r8,%rax,8), %k0, %zmm1	6.147s
131	c[i][j] = c[i][j] + a[i][k] * b[k][j];	0x402acc	vprefetch1 0x1000(%rcx,%rax,8)	11.817s
132	}	0x402ad4	vmovapd 0x40(%r8,%rax,8), %k0, %zmm2	6.982s
133	}	0x402adc	vprefetch0 0x400(%rcx,%rax,8)	9.000s
134	}	0x402ae4	vmovapd 0x80(%r8,%rax,8), %k0, %zmm3	6.596s
135	#endif // }	0x402aec	vprefetch1 0x1000(%r8,%rax,8)	13.835s
136	}	0x402af4	vmovapd 0xc0(%r8,%rax,8), %k0, %zmm4	6.917s
137	}	0x402afc	vprefetch0 0x400(%r8,%rax,8)	8.486s

Selected 1 row(s): 528.3
Selected 0 row(s):

No filters are applied. Process: Any Process Thread: Any Thread Module: Any Module Utilization: Any Utilization

Call Stack Mode: Only user functions Inline Mode: on Loop Mode: Functions only

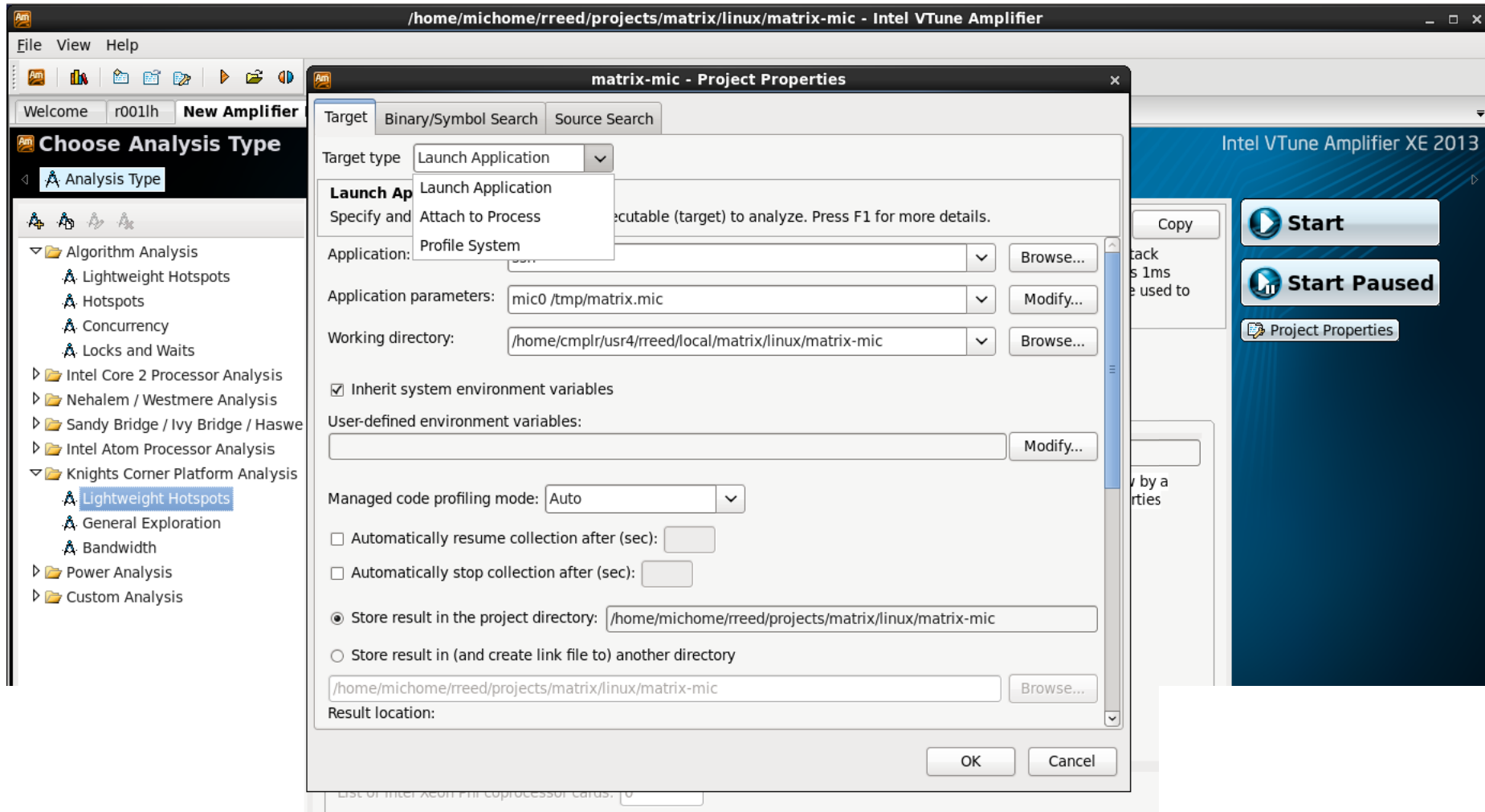
For event collection the coprocessor is treated as a special HW architecture

The screenshot shows the Intel VTune Amplifier XE 2013 interface. The title bar indicates the path: /home/michome/rreed/projects/matrix/linux/matrix-mic - Intel VTune Amplifier. The main window has a menu bar (File, View, Help) and a toolbar. The 'Choose Analysis Type' dialog is open, showing a tree view on the left with categories like Algorithm Analysis, Intel Core 2 Processor Analysis, Nehalem / Westmere Analysis, Sandy Bridge / Ivy Bridge / Haswell, Intel Atom Processor Analysis, Knights Corner Platform Analysis, Power Analysis, and Custom Analysis. Under 'Knights Corner Platform Analysis', 'Lightweight Hotspots' is selected. The main panel displays the 'Lightweight Hotspots - Knights Corner Platform' analysis type. It includes a description: 'Identify your most time-consuming source code. Unlike Hotspots, Lightweight Hotspots has lower overhead when stack collection is disabled. Reduced overhead makes it possible to set a lower sampling interval than Hotspots (as low as 1ms without stacks), which is useful for locating small functions that are called frequently. This analysis type can also be used to sample all processes on a system. Press F1 for more details. Press F1 for more details.' Below this is a text input field for 'List of Intel Xeon Phi coprocessor cards:' with the value '0'. There is an unchecked checkbox for 'Analyze user tasks'. A 'Details' section shows 'Events configured for CPU: Intel(R) Xeon(R) E5 processor' and a note: 'NOTE: For analysis purposes, Intel VTune Amplifier XE 2013 may adjust the Sample After values in the table below by a multiplier. The multiplier depends on the value of the Duration time estimate option specified in the Project Properties dialog.' A table lists events with columns 'Event Name', 'Sample After', and 'Event Description':

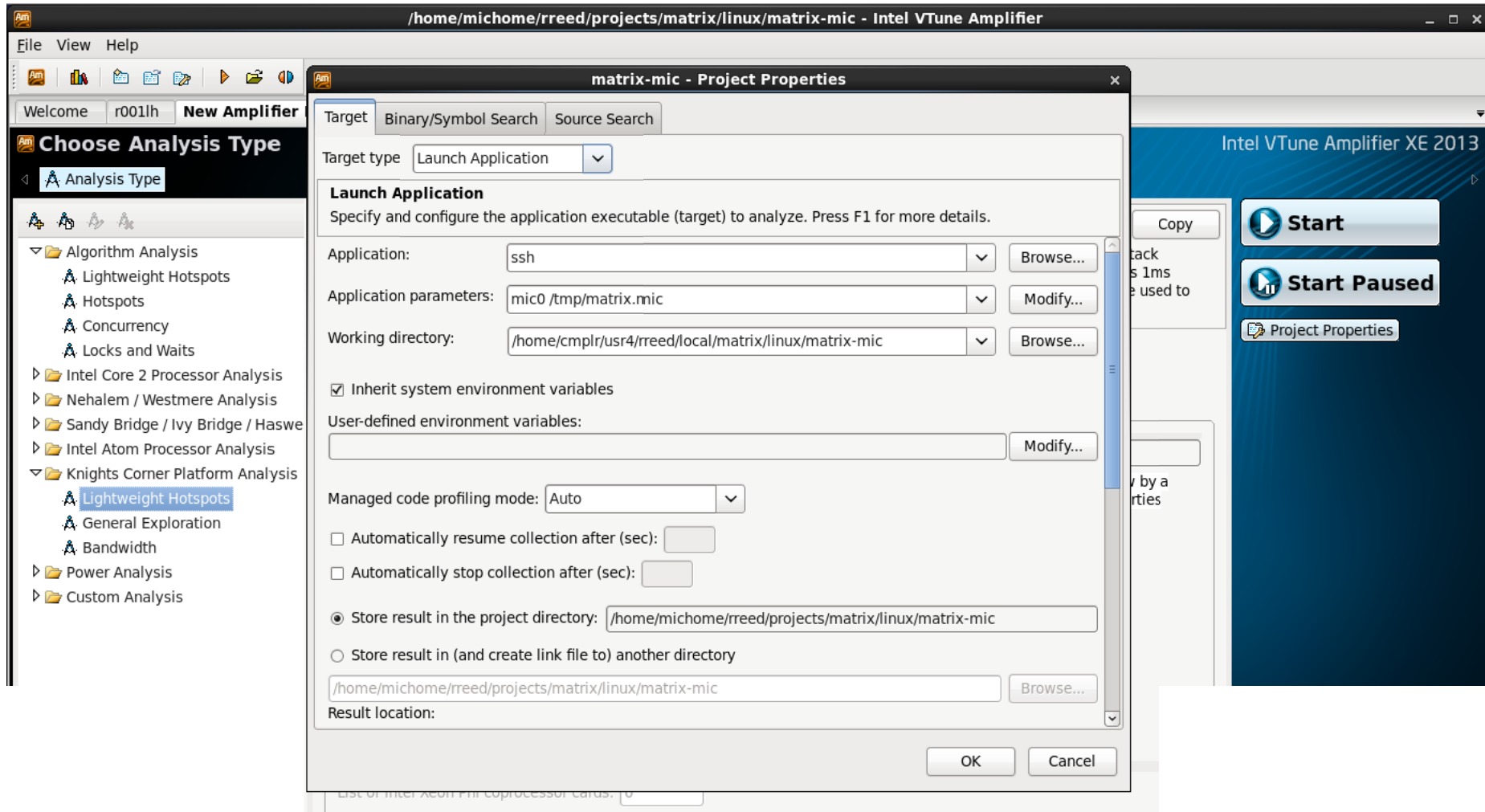
Event Name	Sample After	Event Description
CPU_CLK_UNHALTED	10000000	
INSTRUCTIONS_EXECUTED	10000000	

On the right side of the dialog, there are buttons for 'Start', 'Start Paused', and 'Project Properties'.

Project properties provides the means to invoke data collection by target type

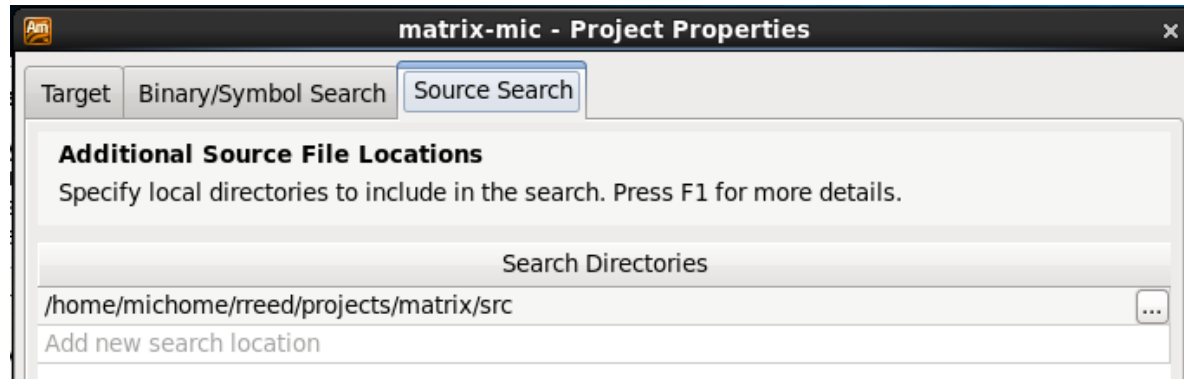


Launch Application serves many uses, from host/offload to native execution



Search directories have been reorganized to speed symbol resolution during finalization

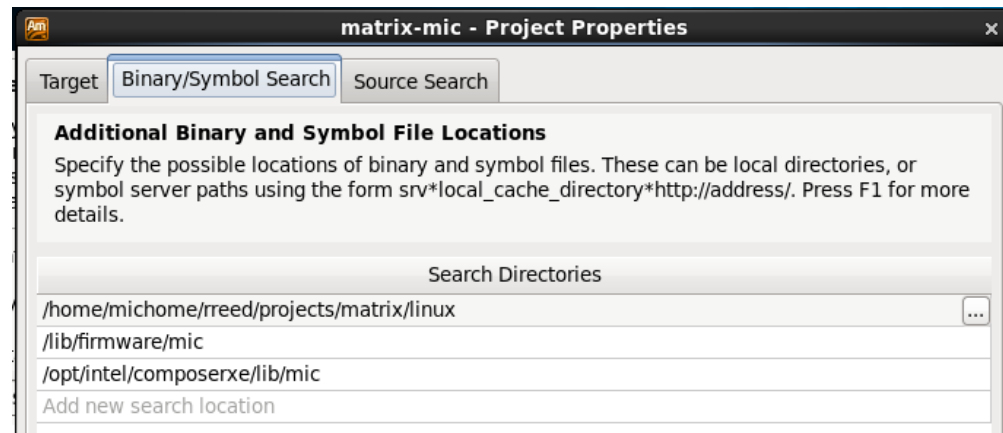
- Enumerate source directories under this tab



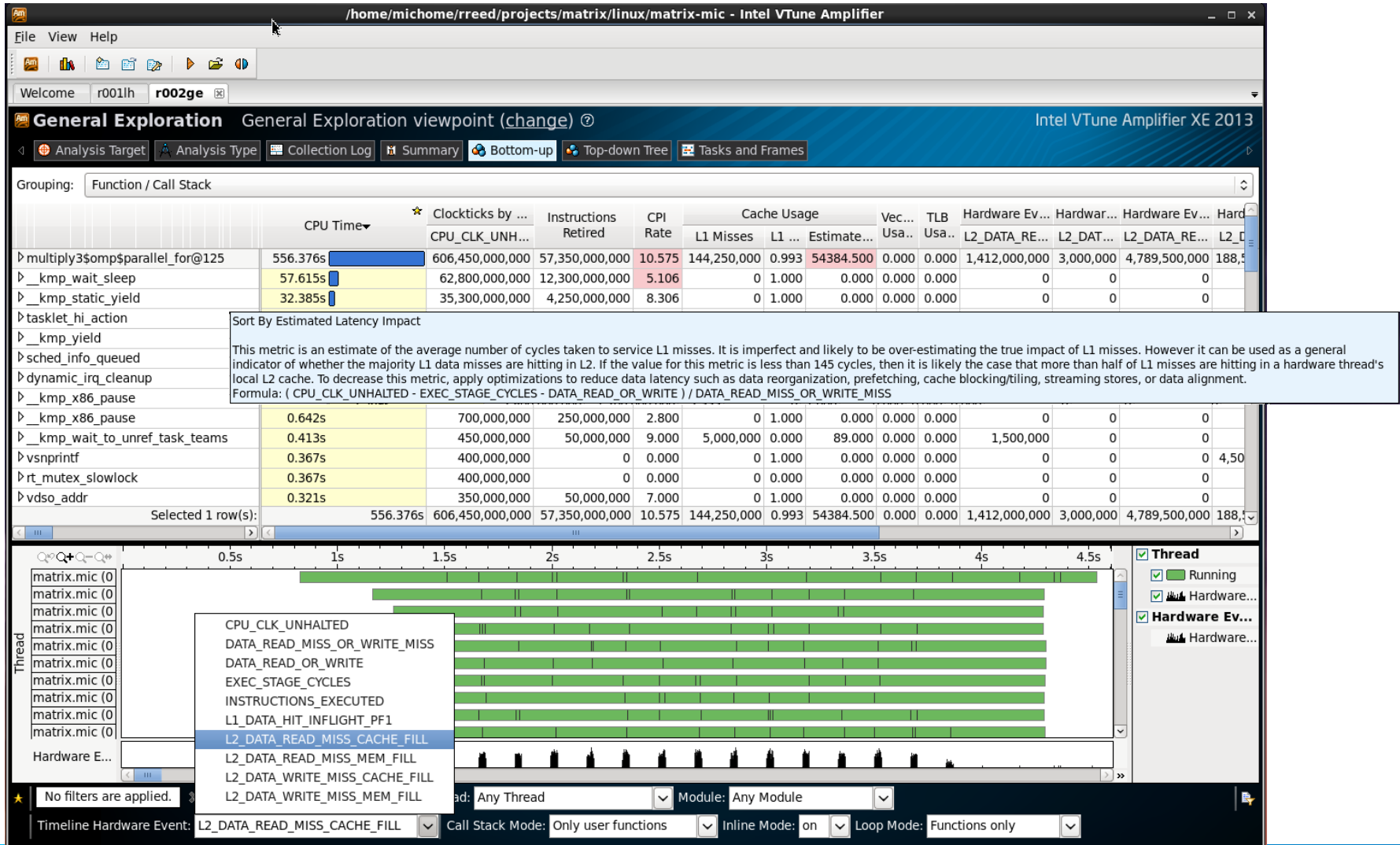
- Put library paths here

Notable coprocessor library paths:

/lib/firmware/mic
/usr/linux-k10m-4.7/linux-k10m/lib64
/opt/intel/composerxe/lib/mic
/opt/intel/composerxe/tbb/lib/mic
/opt/intel/composerxe/mkl/lib/mic
/opt/intel/mpi-rt/4.1.0/mic



General Exploration runs a set of events to drive top-down analysis



Agenda

Start tuning on host

Overview of Intel® VTune™ Amplifier XE

Efficiency metrics

Problem areas

Cycles Per Instruction (CPI), a standard measure, has some special kinks

- Threads on each Intel® Xeon™ Phi core share a clock
 - If all 4 HW threads are active, each gets 1/4 total cycles
- Multi-stage instruction decode requires two threads to utilize the whole core – one thread only gets half
- With two ops/per cycle (U-V-pipe dual issue):

Threads per Core	Best CPI per Core	Best CPI per Thread
1 x	1.0	= 1.0
2 x	0.5	= 1.0
3 x	0.5	= 1.5
4 x	0.5	= 2.0

- To get thread CPI, multiply by the active threads

As an efficiency metric, CPI must be considered carefully: it IS a ratio

- Changes in CPI absent major code changes can indicate general latency gains/losses

Metric	Formula	Investigate if
CPI per Thread	$\text{CPU_CLK_UNHALTED} / \text{INSTRUCTIONS_EXECUTED}$	> 4.0, or increasing
CPI per Core	$(\text{CPI per Thread}) / \text{Number of hardware threads used}$	> 1.0, or increasing

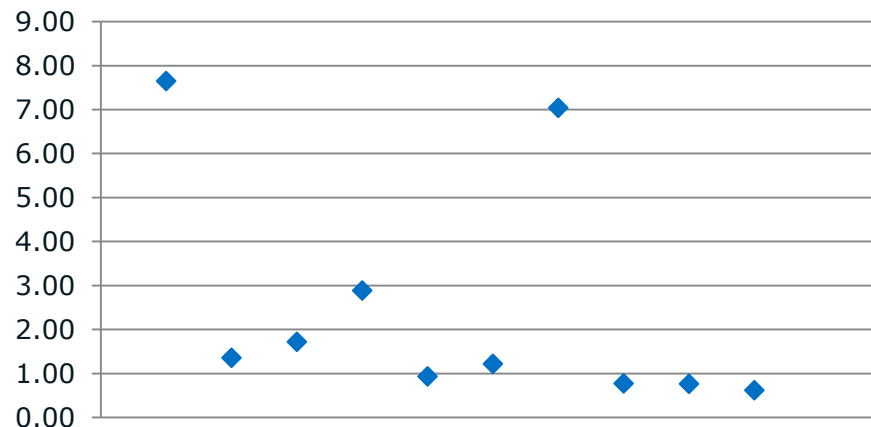
- Note the effect on CPI from applied optimizations
- Reduce high CPI through optimizations that target latency
 - Better prefetch
 - Increase data reuse through better blocking

Two more examples why absolute CPI value is less important than changes

- Scaling data from a typical lab workload:

Metric	1 hardware thread / core	2 hardware threads / core	3 hardware threads / core	4 hardware threads / core
CPI per Thread	5.24	8.80	11.18	13.74
CPI per Core	5.24	4.40	3.73	3.43

- Observed CPIs from several tuned workloads:



Efficiency Metric: Compute to Data Access Ratio

- Measures an application's computational density, and suitability for Intel® Xeon Phi™ coprocessors

Metric	Formula	Investigate if
Vectorization Intensity	$\text{VPU_ELEMENTS_ACTIVE} / \text{VPU_INSTRUCTIONS_EXECUTED}$	
L1 Compute to Data Access Ratio	$\text{VPU_ELEMENTS_ACTIVE} / \text{DATA_READ_OR_WRITE}$	< Vectorization Intensity
L2 Compute to Data Access Ratio	$\text{VPU_ELEMENTS_ACTIVE} / \text{DATA_READ_MISS_OR_WRITE_MISS}$	< 100x L1 Compute to Data Access Ratio

- Increase computational density through vectorization and reducing data access (see cache issues, also, DATA ALIGNMENT!)

Agenda

Start tuning on host

Overview of Intel® VTune™ Amplifier XE

Efficiency metrics

Problem areas*

*tuning suggestions requiring deeper understanding of architectural tradeoffs and application data handling details are highlighted with this “ninja” notation



Problem Area: L1 Cache Usage

- Significantly affects data access latency and therefore application performance

Metric	Formula	Investigate if
L1 Misses	$\text{DATA_READ_MISS_OR_WRITE_MISS} + \text{L1_DATA_HIT_INFLIGHT_PF1}$	
L1 Hit Rate	$(\text{DATA_READ_OR_WRITE} - \text{L1 Misses}) / \text{DATA_READ_OR_WRITE}$	< 95%

- Tuning Suggestions:

- Software prefetching
- Tile/block data access for cache size
- Use streaming stores



If using 4K access stride, may be experiencing conflict misses



Examine Compiler prefetching (Compiler-generated L1 prefetches should not miss)

Problem Area: Data Access Latency

- Significantly affects application performance

Metric	Formula	Investigate if
Estimated Latency Impact	$\frac{(\text{CPU_CLK_UNHALTED} - \text{EXEC_STAGE_CYCLES} - \text{DATA_READ_OR_WRITE})}{\text{DATA_READ_OR_WRITE_MISS}}$	>145

- Tuning Suggestions:

- Software prefetching
- Tile/block data access for cache size
- Use streaming stores



Check cache locality – turn off prefetching and use CACHE_FILL events - reduce sharing if needed/possible



If using 64K access stride, may be experiencing conflict misses

Problem Area: TLB Usage

- Also affects data access latency and therefore application performance

Metric	Formula	Investigate if
L1 TLB miss ratio	$\text{DATA_PAGE_WALK} / \text{DATA_READ_OR_WRITE}$	$> 1\%$
L2 TLB miss ratio	$\text{LONG_DATA_PAGE_WALK} / \text{DATA_READ_OR_WRITE}$	$> .1\%$
L1 TLB misses per L2 TLB miss	$\text{DATA_PAGE_WALK} / \text{LONG_DATA_PAGE_WALK}$	$> 100x$

- Tuning Suggestions:

- Improve cache usage & data access latency
- If L1 TLB miss/L2 TLB miss is high, try using large pages



For loops with multiple streams, try splitting into multiple loops



If data access stride is a large power of 2, consider padding between arrays by one 4 KB page

Problem Area: VPU Usage

- Indicates whether an application is vectorized successfully and efficiently

Metric	Formula	Investigate if
Vectorization Intensity	$\text{VPU_ELEMENTS_ACTIVE} / \text{VPU_INSTRUCTIONS_EXECUTED}$	<8 (DP), <16(SP)

- Tuning Suggestions:
 - Use the Compiler vectorization report!
 - For data dependencies preventing vectorization, try using Intel® Cilk™ Plus #pragma SIMD (if safe!)
 - Align data and tell the Compiler!
 - Restructure code if possible: Array notations, AOS->SOA

Problem Area: Memory Bandwidth

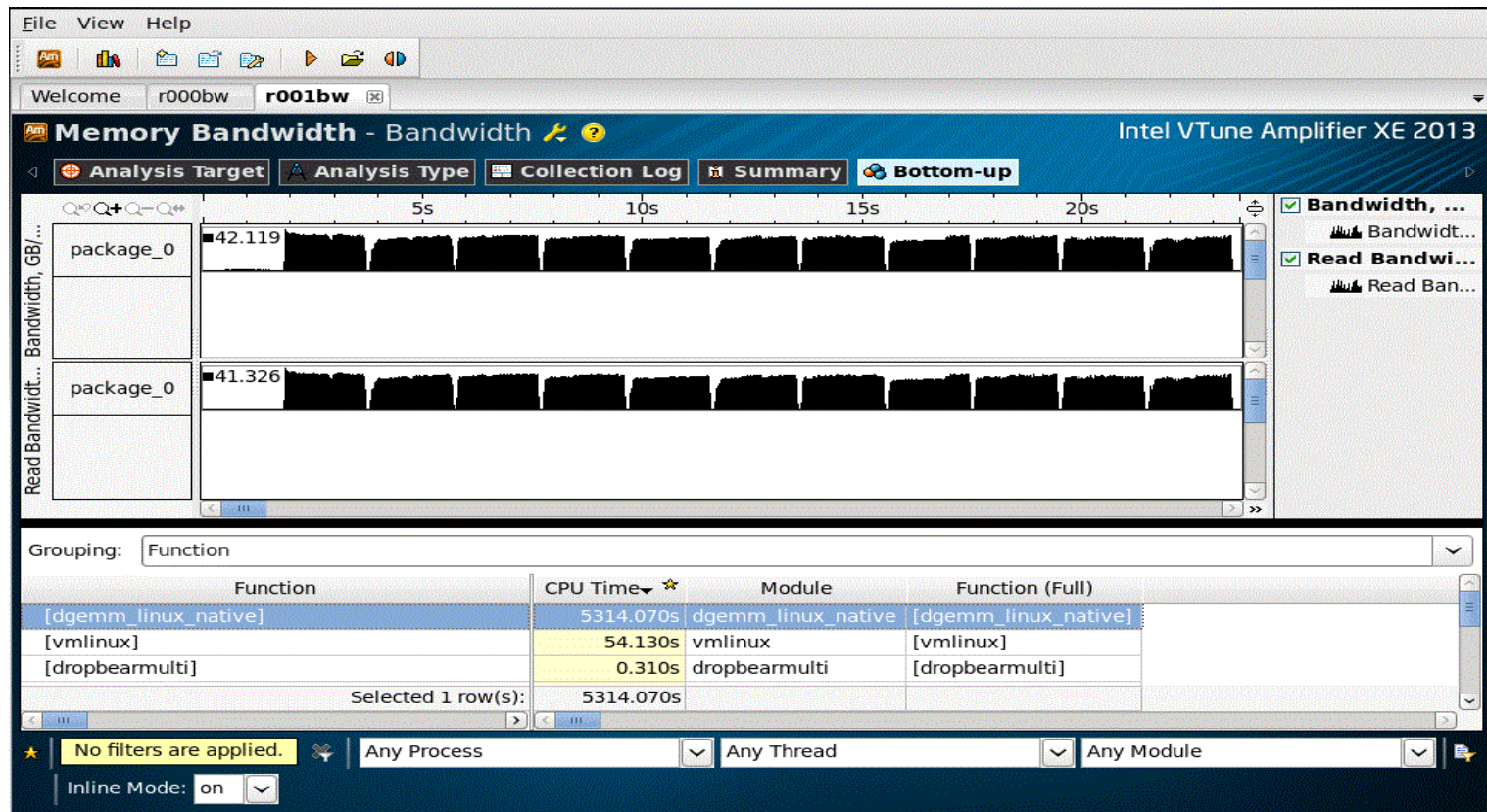
- Can increase data latency in the system or become a performance bottleneck

Metric	Formula	Investigate if
Memory Bandwidth	$(\text{UNC_F_CH0_NORMAL_READ} + \text{UNC_F_CH0_NORMAL_WRITE} + \text{UNC_F_CH1_NORMAL_READ} + \text{UNC_F_CH1_NORMAL_WRITE}) \times 64 / \text{time}$	< 80GB/sec (practical peak 140GB/sec) (with 8 memory controllers)

- Tuning Suggestions:
 - Improve locality in caches
 - Use streaming stores
 - Improve software prefetching

Final caution: coprocessor collections can generate dense volumes of data

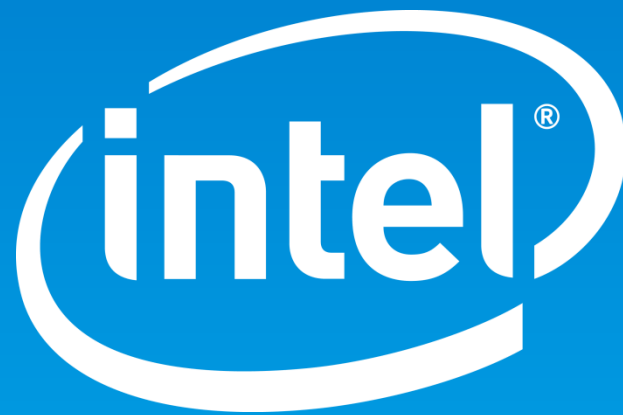
Example: DGEMM on 60+ cores



Tip: Use a CPU Mask to reduce data volume while maintaining equivalent accuracy.

Summary

- Vectorization, Parallelism, and Data locality are critical to good performance for the Intel® Xeon Phi™ Coprocessor
- Event names can be misleading – we recommend using the metrics given in this presentation or our tuning guide at <http://software.intel.com/en-us/articles/optimization-and-performance-tuning-for-intel-xeon-phi-coprocessors-part-2-understanding>
- Intel® VTune™ Amplifier XE supports collecting all of the above metrics, as well as providing special analysis types like General Exploration and Memory Bandwidth



Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Copyright © , Intel Corporation. All rights reserved. Intel, the Intel logo, Xeon, Xeon Phi, Core, VTune, and Cilk are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804